How I installed fltk(aka full-tick) in Microsoft Visual Studio 2019 on Windows 10 pro; And getting the " A display model" from programming Principles and Practices(second edition) up and running. This was done on the current version of Windows 10 pro. Using:

Microsoft Visual Studio code 2019 community edition (version 16.8.4).

Winsdk10.0.18362

fltk-1.3.5

PPP2(tar ball) from stroustrup.com

This is a step by step discription of how I installed fltk1.3.5 in Visual Studio on windows 10.

The first set of instructions is for fltk only. The second set is for Programming Principles and Practices second addition by Bjarne Stroustrup. It is amost identical with exception of debugging and linking the header a source files and creating his first "Hello World Window" from his source files.

For Programming Principles and Practices skip step 14 (generating the fltk "Hello World!" window). The instruction's pick back up for PPP below. Every thing is the same with the exception of debuging the PPP header and source file's. and generating the first window.

1.) Install Microsoft Visual Studio community edition (google it). Choose Desktop Developement with C++ from the Workloads options menu. It should install every thing you need.

2.)Download and install 7zip file manager(it's free and easy to use). Or any other you prefer.

3.)Download fltk1.3.5 from fltk.org(google it).

4.)create a folder some were, I created a folder called dev in users/'my directory':

example C:\users\'name'\(create a new folder here called)dev or something.

5.)go to were you downloaded the fltk tar file (Probably in the downloads folder this is the defualt location on windows).

6.)select the tar or zip file and right click and choose from the menu open with 7zip file manager. 7zip will give you the option of were to unzip and install the file. select browse and navigate to your newly created dev/ folder and select extract here. There may be another tar file extracted to your dev folder. If so select the newly created tar file and right click and select open with 7zip and select extract here. You should now have a fltk-1.3.5 file.

7.)Open this file (fltk-1.3.5) and navigate to IDE\VisualC2010\fltk.sln select it, right clic and choose open with Microsoft Visual Studio 2019. Visual Studio will load all of the project files in this directory, you will see a window asking if you want to update the tool chain(yes) . The ribbon menu should indicate that you are in debug win32 as this should be the defualt. When its finished updating the tool chain, in the solutions explorer window there is a list of projects, the project Demo should already be highlighted if not select it and select from the ribbon menu build/build solution or press f5.

there are a lot of projects that wil be built and a lot of cmd windows will pop up during the build. Just let it finish. When its done there will be an fltk\test\ demod.exe in the fltk-1.3.5\test folder were you put fltk, that you can use to explore fltk with. You can sift through these at this time or come back to them later. Lets finish the install first.

8.)Once Visual studio is done and all files are compiled. select save all from the file menu on the ribbon tab. and close Visual Studio code 2019.

9.)open file explorer in windows 10 and navigate to were you put fltk-1.3.5, for me that is

c:/users/'myname'/dev/fltk-1.3.5.

open the file and navigate to the lib folder and select each lib file, one at a time or for myself it was simpler to copy all the files by highlighting or using select all and choosing copy.

then navigate back to c:\Program (x86)\Microsoft Visual Studio\2019\Community\VC\Tools\MSVC\14.28.29333(this may be slightly different depending on the version)\lib\x86 do not open the x86\ folder when you get here just select(so it's highlighted) it and right clic and select paste. The dialog box will ask for permision check the box for all files and click yes. Once you have copied the files (at this point should still only have debug libraries)open the lib\86 folder scroll to and delete the fltk.bcs files. they don't actually belong there. Also the read me file if you happened to copy it as well. Leave all of the fltk debug files there are 7 total. and be carefull not to delete any thing else. We are only interested in the fltk debug files at this time. You will recognize them becuase they all start with fltk and end with d.lib.

10.)now open fltk-1.3.5 again and navigate to and copy the FL folder(don't open it just select and copy the whole folder). Navigate back to your previous location at ...\MSVC\14.28.29333\(this time locate and highlight the include folder right clic and select paste, select do this for all from the permissions box and it will copy FL to include. Repeat the process for the GL folder in fltk-1.3.5 to visualstudio include folder.

11.) Locate the fluid folder in fltk-1.3.5 open it and select and copy fluid .exe or you may only have fluidd for now and navigate back to the same \MSVC\14.28.29333\(select the bin folder and paste fluidd and or flluid.exe into the bin folder.

restart Microsoft Visual Studio Code 2019.

12.)When VScode reopens select create a new project and from the menu that opens select empty project. Give it a name I named mine fltk32. select next and a empty project page will open. Note: if your just setting up Programming Principles and Practices name the project win32 that's whats asked of you to do in the book. Really the book ask you to create a empty win32 project. It defualts to x86 but it does not seem to be a problem. I checked the box to place the solution in the same directory (optional).

13.) on the ribbon tab select Project and navigate to properties at the bottom of the drop down menu. Open the proprties pages and select linker/input. On the input menu on the line: Additional Dependencies select edit. Enter the following lib's, seperating each one by a simicolon.

comctl32.lib;wsock32.lib;fltkd.lib;fltkformsd.lib;fltkgld.lib;fltkimagesd.lib;fltkjpegd.lib;fltkpngd.lib;user32.lib;gdi32.lib;advapi32.lib;shell32.lib;ole32.lib;

since this is for installing fltk in VS I am including all 7 debug libraries. The Windows lib are nesssacary in order to create a windows executable window that will launch the fltk window.

In Windows a window has to be registered and given a handle before being handed over to the system. This will link fltk and windows.

new to visual studio is avoiding conflicts between libraries of the debug and release types. So initialy I got a warning about potential conflicts all thow just a warning it seemed prudent to avoid conflicts so the next step was added after cunsulting the information on the windows linker warnings page of the microsoft documents pages you can find more information at. https://docs.microsoft.com/en-us/cpp/error-messages/tool-errors/linker-tools-warning-lnk4098?view=msvc-160.

14.) In the same linker\input page from the project properties tab select Ignore Specific Defualt Libraries and select edit. In the Edit box enter.

/NODEFUALTLIB:[libcd.lib,libcmt.lib,msvcrt.lib,libcmtd.lib]

select apply. then ok.

These are the release lib and we are using debug so when you do eventually do a release version you set up the same procecdure but instead ignore the debug libraries and change the additional dependencies to the release versions of the fltk lib after you build a release version.

Note: There is no compiler tab yet in the Project properties settings because ther is nothing to compile yet. After adding the source files before you compile come back to this next step and check to see that it's sep up correctly.

15.) From the same Project Properties menu select c/c++ linker and select code generation and from that menu select Multi-threaded Debug DLL (/MDd) Again for debug mode you may have to change this for a release version as well. The line: runtime checks should be set to Default.

16.)At this point the compiler and linker should be set up right. Now let's create a source file. If your working on programming principles and practice's skip this next step.

The file fltk-1.3.5 does not contain a "Hello World!" window which is the custom. However the fltk user manual does wich you can download from fltk.org if you want to learn more about fltk in general. I have included that file to use as a first source file for fltk.

In the Solution Explorer window select from the drop down menu Source Files and right clic and select add new item. From the pop up menu select visual c++ /C++ file(.cpp) in the editor box copy and paste the following code.

```
#include <FL/Fl_Window.H>

#include <FL/Fl_Box.H>

int main(int argc, char** argv) {

        Fl_Window* window = new Fl_Window(340, 180);

        Fl_Box* box = new Fl_Box(20, 40, 300, 100, "Hello, World!");

        box->box(FL_UP_BOX);

        box->labelfont(FL_BOLD + FL_ITALIC);

        box->labelsize(36);

        box->labeltype(FL_SHADOW_LABEL);

        window->end();

        window->show(argc, argv);

        return Fl::run();

}
```

```
from the Top of the editor window or the Properties Explorer window select win32
and from the ribbon menu select build or press f5.

you should see a succesfull build in the output window.

from the ribbon menu choose Tools/Command line: which ever you prefer cmd shell or
powershell

> cd debug (The executable file will be located in the debug folder).

enter win32.exe you should get the fltk "Hello World!" window.

congradulations if all went as it should you have linked fltk and windows.


Programming Principles and Practices (second edition)

if you have not already done the first 10 steps of the previous instructions do so
now.
```

Instruction's for Programming Principles and Practice's(2 edition):

17.) You will need the header and cpp source files from Bjarn Stroustrup's web page here is a link to the support code for PPP2(tar ball).

https://www.stroustrup.com/programming_support.html

Note that it say's the code has been recovered from his previous website and may contain bugs. And it does. It was no easy task finding fixes searching the verious sites like stack exchange gethub, you tube, Windows, and others for pehaps original files that were used before the recovery and find fixes. And more than once thought about skipping these chapters all together. But through a combination of a lot of incomplete answers I came up with this solutuion which is also incomplete but it does get them installed and working thow there will be some warnings a lot of witch can be ignored and some are due to the quirkyness of visual studio code's need for fully qualified names in some area's. One of the main themes of this book is to place an emphasis on debugging so it will be up to you the learner to handle what bugs may still exist.

18.) Download the PPP2(tar ball) and create a folder for it some were. Use 7zip to extract the files(or the extraction tool that you use). It may be better to edit the file's and correct the bugs before using it in visual studio. Navigate to the PP2 folder/file and open in a text editor(note pad will do but it helps to have line numbers)or which ever one(editor) you use.

It's not necassary to read through this section. It only serves to explain the reasoning behind this solution. You may skip to the next step. I included this in-case someone would like to point out a more correct solution.

In chapter 12 of PPP(second edition) there is a graphic that shows the way the file's are linked:

we have:

int main()->"Graph.h",Simple_window.h"

"Graph.h"->"Point.h","fltk.h"

"fltk.h"<-fltk code(The arrow i believe should point the other way, printing error ?)

"Simple_Window.h"->"Gui.h","Window.h"

"Gui.h"->"Window.h","fltk.h"

"Window.h"->"fltk.h","Point.h"

"fltk.h"<-fltk code(I believe the header calls the fltk code and all needed fltk source files)

"Gui.cpp"->"Gui.h"

"Graph.cpp"->"Graph.h"

"window.cpp"->"Window.h"

The code from the PPP2(tar ball) stack look's like:

main()->"Graph.h","Simple_window.h"

"Graph.h"->"Point.h,"fltk.h",<vector>

"fltk.h"->"fltk code

"Simple_window.h"->Gui.h

Gui.h->"Point.h","fltk.h","Window.h","Graph.h"

"Window.h"->"fltk.h","std_lib_facilities.h",Point.h

"window.cpp->"Window.h","Graph.h","Gui.h"

"Gui.cpp->"Gui.h","std_lib_facilities.h",<sstream>

"Graph.cpp"->"Graph.h",<map>

This clearly is not what is described in chapter. It's more like a circle then a stack. There are source and header files out of place and creating conflicts with definitions as well as some that will not find the definition or declaration they need.

So after a lot of hair pulling, nail biting , wincing, a few "I want my mommie" , I managed to come up with a working model that look's like this:

int main()->"Graph.h","Simple_window.h"

"Graph.h"->"std_lib_facilities.h",Point.h","fltk.h"

"Simple_window.h"->"Gui.h"

"Gui.h"->"Window.h",Graph.h"

"Window.h"->"std_lib_facilities.h","Point.h","fltk.h"

"std_lib_facilities.h"->stdlib <headers>(moved map here, out of Graph.cpp)

"fltk.h"->fltk code

It is not clear if this is entirely accurate and it also is not what is described in the chapter. Comparing the two reveals that this is a more direct and ordered stack from the bottom tier to top tier.

Top tier being "std_lib_facilities","Point.h", and "fltk.h"

the middle tier ("Graph.h", and "Windows.h) are the only headers that call the top tier but do not call each other)and the bottom tier must go through "Graph.h" or "Windows.h" . Also the standard library headers can all be called from one source(std_lib_facilities) which helps to orginize things.

18.) Lets start with Simple_window.h: Select it from the PPP2 folder and open in a text editor. Note: I used #pragma once on the first line at the top of each file. This may not work with every compiler and may not be needed. If you need to know more you can google it. And decide if you want to include it. I believe VS will use it on some files by default.

#pragma once

#include "Graph.h" (need's to be inluded).

change "GUI.h" to "Gui.h"

edit the line: struct Simple_window: Window { ....}

struct Simple_window: Graph_lib::Window{....}

select File/save changes.

Since Simple_window.h includes GUI.h and Graph .h we will do them next.

select Graph.h open in the editor.

#pragma once

#include "Point.h"

#include "fltk.h"

#include"std_lib_facilities.h"

comment out all other header files present (example. //#include <vector>)

select save and close.

select Gui.h from the PPP2 file, In the text editor window make the following change's.

#pragma once(line 1)

comment out all header's except "Window.h" and "Graph.h"

around line 92(line numbers may vary from editor to editor) change the following line comment out s and write in label. I am cautious about deleting anything in case i need to change it back. Add the constructor.

```
Menu(Point xy, int w, int h, Kind kk, const string& s); <-(remove that simi colon
syntex error) and change too.

Menu(Point xy, int w, int h, Kind kk, const string& /*s*/ label)

       :Widget(xy, w, h, label, 0), k(kk), offset(0){}

around line 112:

 void attach(Window& win)

       {

              for (int i=0; i<selection.size(); ++i) win.attach(selection[i]);

              own = &win;

       }      (add own=&win;)
```

save and close.

select Point.h from PPP2 and open in the text editor window change the following

uncomment everything (all the lines of code that have been commented out).

add #pragma once on first line before anything else.

save and close.

select "Windows.h"

#pragma once (Top line before anything else)

#include "fltk.h"

#include "std_lib_facilities.h"

#include "Point.h"

comment out all other includes.

save and close.

select Graph.cpp from source in the Properties Exporer and make the following changes in the text editor.

#include "Graph.h"

comment out all other includes.

change the line (around 312 -> 315 will vary but it is the bool can_open(....) function) from: return ff;

change to: return bool(ff);

save and close.

select Gui.cpp from PPP2 and in the text editor change:

comment out all include's except "Gui.h"

comment out:

Menu::Menu(....){} around line 55. This function has already been defined .

it has already been constructed and defined in anoter location.

save as to Projects folder

select "std_lib_facilities.h from the Solution explorer and in the editor window change:

#pragma once (on line 1)

below pragma but before any thing else add:

```
#ifndef STD_LIB_FACILITIES_H_INCLUDED

#define STD_LIB_FACILITIES_H_INCLUDED

#include<map> (add to the list of includes)

add on the last line:

#endif // STD_LIB_FACILITIES_H_INCLUDED
```

Around line 222 -> 237(line numbers may vary so it's near there) comment out the default_random_engine& get_rand(...){}

and the helper functions int randint(),seed randint(..),int randint(max).

This function is apparently creates conflicts in some other header and source file's.

select save and close.

Open Visual Studio code 2019 community edition.

Open Visual Studio Code and if you have not done so at this time select create a new poject. From the menu select emty project and name it win32. I have checked the box to save the solution to the same project folder this is optional.

19.)In the Solution explorer(if not already open you can open it by selecting from the ribbon menu View/Solution explorer) of your project select Headers from the drop down menu and right clic and select add existing item. copy the header files from ppp2 into Header files folder. You will need fltk.h, Graph.h,Gui.h, Point.h, Simple_window.h, Window.h, and std_lib_facilities.h. A note on std_lib_facilities.h in the chapter introduction it say's that you can use this header file or start dealing directly with the std libraries. I am using the std_lib..header here it has all the needed std facilities that more than one header file depends on. If you choose not to use it at least use it here until things are up and running.

use the same procedure to copy the cpp files to the source folder. You will need Graph.cpp, Gui.cpp, and Window.cpp. We intend to build and link these file's all in one go but the compiler but the linker will still need to find them. So we will copy them one by one to the projects folder.

After making the corrections to these files select them one at a time and from the ribbon menu select File/save as: navigate to were VS has saved your project, the defualt location should be a folder called source/repos/the name of your project. Open the project folder and save the corrected header and source files. This will allow the linker to find them.

now go back to step 15 and check the c/c++ settings.

Go back to step 13 and 14 and set the linker settings

once done continue with step 18.

18) Add the main() entry point cpp:

From the Solution Explorer window select source files and right clic. and select add new item. From the pop up menu select visual c++ /C++file(. cpp).Name the file, I named mine simple_window or leave it as source.cpp if you prefer then select add. In the new blank editor box create the cpp file from PPP(2 edition) chapter on "A Display Model. Here is the first part of that file you can use it or use the book. if you use this copy you should still read the chapter. This is not the complete chapter source code. But it should create an executable.

```
#include "Simple_window.h"

#include "Graph.h"


using namespace Graph_lib;


int main()


      try
```

```
{

        Point tl{ 100,100 };
        Simple_window win{ tl,600,400,"Canvas" };
        Graph_lib::Polygon poly;
        poly.add(Point{ 300,200 });
        poly.add(Point{ 350,100 });
        poly.add(Point{ 400,200 });
        poly.set_color(Color::red);
        poly.set_style(Line_style(Line_style::dash, 4));
        win.attach(poly);
        win.wait_for_button();

        Axis xa{ Axis::x,Point{20,300},280,10,"x axis" };
        //make an axis, axis is a kind of shape
        // x axis, means horizontal starting at (20,300), 280 pixals long
        //10 "notches, label it x axis
        win.attach(xa);
        win.set_label("Canvas #2");
        win.wait_for_button();

        Axis ya{ Axis::y,Point{20,300},280,10,"y axis" };
        ya.set_color(Color::cyan);
        // choose a color
        ya.label.set_color(Color::dark_red);
        // choose a color for the text
        win.attach(ya);
        win.set_label("Canvas #3");
        win.wait_for_button();


        Function sine{ sin,0,100,Point{20,150},1000,50,50 };

        //sine curve plot sin() in range [0:100] with (0,0)at (20,150)

        //using 1000 points; scale x values *50,scale y values *50

        win.attach(sine);

        win.set_label("Canvas #4");

        win.wait_for_button();


        Graph_lib::Rectangle r{ Point{200,200},100,50 };

        //top left corner,width,height

        r.set_fill_color(Color::yellow);

        win.attach(r);

        win.set_label("Canvas #6");

        win.wait_for_button();


        Closed_polyline poly_rect;

        poly_rect.add(Point{ 100,50 });

        poly_rect.add(Point{ 200,50 });

        poly_rect.add(Point{ 200,100 });

        poly_rect.add(Point{ 100,100 });

        poly_rect.add(Point{ 50,75 });

        poly_rect.set_fill_color(Color::green);

        win.attach(poly_rect);

        win.set_label("Canvas #7");

        win.wait_for_button();


        Text t{ Point{150,150},"Hello Graphical World!" };
```

```
        win.attach(t);

        win.set_label("Canvas #8");

        win.wait_for_button();


        t.set_font(Font::times_bold);

        t.set_font_size(20);

        win.set_label("Canvas #9");

        win.wait_for_button();


        Image ii{ Point{100,50},"image.jpg" };

        // 400*212-pixel jpg

        win.attach(ii);

        win.set_label("Canvas #10");

        win.wait_for_button();


}

catch (exception& e) {

        // error reporting

        return 1;

}

catch (...) {

        //more error reporting

        return 2;

}
```

create this file for an entry point as a cpp file. Select from the Solution Explorer win32(or the project name)on the ribbon menu/ build or press f5.

select Tools/Command line and choose the shell that you prefer.

```
>cd debug
```

enter dir (if in cmd window) or ls if in (Power shell) there should be an .exe file.

I hope this works for you if you have problems go back double check that you have completed all the steps in order.

good luck.